Amendments to the Claims

Please amend claims to be as follows.

1. (original) A method of reducing access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor, the method comprising:

   receiving a command to write an interrupt mask value to the task priority register;

   writing the interrupt mask value to the task priority register; and

   writing the interrupt mask value into a shadow copy of the task priority register,

   wherein the shadow copy is written each time that the task priority register is written.

2. (currently amended) The method of claim 1, further comprising:

   receiving a command to read the interrupt mask value from the task priority register; and

   reading the interrupt mask value from the shadow copy, instead of from the task priority register.

3. (original) The method of claim 1, wherein the shadow copy is always written after the task priority register is written.

4. (original) The method of claim 3, further comprising, upon receiving an interrupt, reading the interrupt mask value from the task priority register and writing the interrupt mask value to the shadow copy.

5. (original) The method of claim 1, wherein, if the task priority register is accessed frequently, then the shadow copy is stored in low-latency cache memory within the microprocessor.

6. (original) The method of claim 1, wherein the method obviates a need to use a serialize instruction after the task priority register is written because each interrupt performs a serialize operation and performs a read of an interrupt vector register.

7. (original) The method of claim 6, whereby a latency of writing to the task priority register is substantially reduced.

8. (original) The method of claim 2, whereby a latency of reading from the task priority register is substantially reduced.

9. (currently amended) The method of claim 1, wherein ~~the microprocessor comprises an IPF architecture microprocessor~~ data in the task priority register reflects a level of priority of tasks being performed by the microprocessor.

10. (currently amended) The method of claim 9, wherein ~~the local PIC unit in the microprocessor comprises a streamlined advanced programmable interrupt controller (SAPIC) unit in the IPF architecture microprocessor~~ the task priority register comprises eight bits to designate up to 256 priority states.

11. (original) The method of claim 1, wherein the method is performed by an operating system.

12. (currently amended) The method of ~~claim 9~~ claim 4, ~~wherein the method is performed by a UNIX-type operating system configured for the IPF architecture microprocessor~~ further comprising, after writing the interrupt mask to the shadow copy, reading an interrupt vector register at a beginning of an interrupt handler.

13. (currently amended) The method of ~~claim 9~~ claim 12, ~~wherein the method is performed by a version of a Windows operating system configured for the IPF architecture microprocessor~~ further comprising executing instructions specific to the interrupt handler and returning from the interrupt handler.

14. (currently amended) A computer-readable medium storing an [[An]] operating system with reduced access latency to a task priority register of a local programmable interrupt controller unit within a microprocessor, the ~~operating system~~ computer-readable medium comprising:

microprocessor-executable code configured to write a priority level to the task priority register; and

microprocessor-executable code configured to write the priority level into a shadow copy of the task priority register,

wherein the shadow copy is written each time that the task priority register is written.

15. (currently amended) The ~~operating system~~ computer-readable medium of claim 14, further comprising:

microprocessor-executable code configured to read the priority level from the shadow copy, instead of from the task priority register.

16. (currently amended) The ~~operating system~~ computer-readable medium of claim 14, wherein the shadow copy is always written after the task priority register is written.

17. (currently amended) The ~~operating system~~ computer-readable medium of claim 16, further comprising,

microprocessor-executable code configured to, upon receipt of an interrupt, read the priority level from the task priority register and write the priority level to the shadow copy.

18. (currently amended) The ~~operating system~~ computer-readable medium of claim 14, wherein, if the task priority register is accessed frequently, then the shadow copy is stored in low-latency cache memory within the microprocessor.

19. (currently amended) The ~~operating system~~ computer-readable medium of claim 14, wherein the operating system avoids a need to use a serialize instruction after the task priority register is written because each interrupt performs a serialize operation.

20. (currently amended) The ~~operating system~~ computer-readable medium of claim 19, whereby a latency of writing to the task priority register is substantially reduced.

21. (currently amended) The ~~operating system~~ computer-readable medium of claim 15, whereby a latency of reading from the task priority register is substantially reduced.

22. (canceled)

23. (canceled)

24. (canceled)

25. (canceled)

26. (currently amended) A method of reducing a latency to read a task priority register of ~~an IPF type~~ a microprocessor, the method comprising:

receiving a command to read an interrupt mask value from the task priority register; and

reading the interrupt mask value from the shadow copy at a memory location, instead of from the task priority register itself.

27. (currently amended) An operating system stored on a computer-readable medium with reduced latency to read a task priority register of a local programmable interrupt controller unit within a microprocessor, the operating system comprising microprocessor-executable code configured read the interrupt mask value from the shadow copy at a memory location, instead of from the task priority register itself.

28. (original) A multiple-processor computer system comprising:

a plurality of microprocessors interconnected by a processor bus, wherein each microprocessor includes a task priority register (TPR) with a interrupt mask value for that microprocessor;

a memory system, including local cache memory on each microprocessor and a main memory,

wherein the memory system holds data including an operating system and shadow copies of the TPRs, and

wherein the operating system includes executable-code for reading the interrupt mask values from the shadow copies and for maintaining the shadow copies.

29. (original) A method of reducing latency to write a task priority register within a microprocessor, the method comprising:

upon receiving a command to write an interrupt mask value to the task priority register, writing the interrupt mask value to the task priority register without performing a serialization directly thereafter; and

upon receiving an interrupt, performing the serialization and reading an interrupt vector register, wherein a spurious indicator is returned if the interrupt is maskable.